CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of $75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

APR 05 1996

When renewing by phone, write new due date below previous due date. L162

Digitized by the Internet Archive
in 2013

http://archive.org/details/multirateordinar959orai

I 6r
no. 959
cop. 2

UIUCDCS-R-79-959

UILU-ENG 79 1704
COO-2383-0056

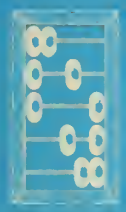
A MULTIRATE ORDINARY DIFFERENTIAL EQUATION INTEGRATOR

by

A. Orailoglu

THE LIBRARY OF THE
APR 17 1979
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

March 1979



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-79-959

A MULTIRATE ORDINARY DIFFERENTIAL EQUATION INTEGRATOR

by

A. Orailoglu

March 1979

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

Supported in part by the U. S. Department of Energy, Grant US ENERGY/EY-76-S-02-2383 and submitted in partial fulfillment of the requirements of the Graduate College for the degree of Master of Science in Computer Science.

510.84
I 2 6 e
no.959-964
cop 2

ACKNOWLEDGMENT

The author wishes to express his gratitude to Prof. C.W. Gear for his invaluable ideas, suggestions, criticisms and supervision. Without them, this thesis could not have been possible. The author would also like to thank every single member of his research group and all his friends for their unfailing help, suggestions and support. Special thanks are due, of course, to his parents, Hristo and Atanasia Orailoglu, for everything they have done, and for this the author would like to acknowledge his great debt to them.

TABLE OF CONTENTS

I - INTRODUCTION..... 1

II - IMPLEMENTATION..... 3

III - RESULTS..... 11

a - Efficiency Comparisons..... 11

b - Parametric Study..... 20

IV - PROGRAM USAGE..... 25

V - FURTHER IMPROVEMENTS..... 27

VI - CONCLUSIONS..... 29

BIBLIOGRAPHY..... 30

APPENDIX..... 31

I - Program..... 31

I. INTRODUCTION

This thesis will concern itself with the development of a package for solving systems of ordinary differential equations using multirate techniques. In a paper by C.W. Gear [1] some of the theoretical questions associated with multirate methods are discussed. Yet most of the work that has been done up to now in solving systems of ordinary differential equations has concerned itself with solving the whole system simultaneously. A disadvantage of solving systems of ode's simultaneously is clear even intuitively. When some dependent variables change much more rapidly than others, the whole system will be forced to take time steps to suit the fastest dependent variable. Coupled with the fact that it is easy to construct examples in which different equations will be slowly moving at different parts of the integration interval, it becomes obvious that to integrate the equations of the system of ode's separately would be advantageous. The idea of integrating the equations of the system of the ode's separately is thus the one that gives rise to multirate methods.

Attractive and easy though it sounds, multirate methods are not simple to implement. When we are solving systems of ordinary differential equations, we usually have systems in which the equations are coupled . Therefore we cannot let each equation proceed as far as its local error estimator would permit. The reason for this becomes clear when we consider that in the system

$\tilde{y}' = f(\tilde{y})$ the behaviour of y_i' depends not only on y_i but on the whole vector $\tilde{y}$. Therefore when we are advancing along the solution curve of y_i , our results, which depend on our estimates of $\tilde{y}$, will only be as good as our estimates of $\tilde{y}$. Our estimate of $\tilde{y}$, will depend , on the other hand, on the solution curve of $\tilde{y}$ and on the length of the step taken.

Because of the peculiarities of the multirate method, we cannot be content with the recommended step that our ode integrator returns. We have to use more careful monitoring and checking of acceptable next steps. An experimental package, MRATE , which uses multirate methods, has been developed based on DIFSUB [2] . In this package, a number of different schemes have been tested. Descriptions of implementation techniques, different schemes tested and experimental results follow.

II. IMPLEMENTATION

The general organization of the scheme is to view the system as n single equations, each being integrated separately. At any intermediate stage , each equation has been integrated to a different time called t_{curr}. This is a vector with a component for each member of the system. A stepsize h has also been estimated for the next step for each equation, leading to the definition of $t_{poss} = t_{curr} + h$.

The next equation to be integrated one step is that with the smallest t_{poss} . During the integration of a particular equation, for example y_i , the function $f_i(y_i;y_j,j\neq i;t)$ must be calculated. The values of $y_j(t)$ are calculated by extrapolation from the last step.

Were it not for error control of stepsizes, the task would be straight-forward. The difficulties arise because of the possibility that the estimated step for an equation may be far too large, so that that equation is not integrated but its correct values are extrapolated too far into the future for use in other equations. If this is not discovered in time, it is virtually impossible to back up the equations which have been integrated with the incorrect values.

The need to monitor time steps suggested by the ode integrator for a single equation of the system of ordinary differential equations has prompted the following additions to

the basic organization.

1 - Since after acceptable projected times ($t_{poss} = t_{curr} + h$) have been calculated, other equations may be integrated, an estimate of the effect of the changes on the acceptability of the projected time steps has been included. If the effects pass a certain acceptability level, the projected time steps are recalculated.

2 - To determine the influence of each integration on the projected time steps of the coupled equations, information on the coupling of the systems is needed. The Jacobian of f is needed since our estimate of the effect depends on it. Also, to economize on the work done by the interpolating routine, information on which system is being influenced by which variable is needed. The reason for this is that usually only a small number of variable values are needed in order to evaluate the equation $y_i' = f_i(\tilde{y};t)$. All this information is presently being requested of the user, but in a more comprehensive software program, this information can probably be deduced by the program itself.

With this information the following implementation was chosen :

1 - As an ode integrator, DIFSUB was used. The integrator was chosen for its ready accessibility and also for its particular ease in extrapolating to different time values. Only the Adams predictor-corrector method (of the three possible options of the program) was used.

Since DIFSUB does not use any multirate integration, but integrates the system simultaneously, extensive modifications occured. At present, the multirate integration package, MRATE, handles only one equation per call instead of the entire system. To facilitate correctness and ease of debugging, the ode integrator was broken down into the following subroutines, as it was being modified.

Scale (to scale information associated with y_i depending on h)

Predct (Predictor)

Corect (Corrector)

Soe (Step and order change mechanism)

Sharp (to reduce to a first order method if integration does not seem to converge)

Update (to update variables associated with order each time order used is changed)

Errchk (to check if converged corrector's error estimate is within the tolerance specified)

Also MRATE now calls the interpolating routine interp before a call to difunl (which evaluates $f_i(\tilde{y};t)$) to supply the values of y through interpolation.

2 - Since DIFSUB saves all known derivatives of each variable at its current time, the natural choice for interpolation was Taylor's expansion.

DIFSUB saves the j-th derivative of y_i scaled by the quantity $h^j/j!$ where h is the next step recommended. Since a Taylor's expansion would be

$$h^* = t_i - t_j$$

$$y_j(t_j + h^*) = \sum_{m=0}^{\infty} \frac{(h^*)^m y_j^{(m)}(t_j)}{m!} \simeq$$

$$\simeq \sum_{m=0}^{k} \frac{(h^*)^m y_j^{(m)}(t_j)}{m!}$$

it follows that all we need in order to interpolate to the t_i point is rescaling the information saved in DIFSUB. If s_m is the saved information for the m-th derivative in DIFSUB, our interpolating procedure produces:

$$y_j(t_j + h^*) = \sum_{m=0}^{k} \left[\frac{h^*}{h}\right]^m s_m$$

Within this structure, it was easy to test several schemes for choosing which variable to integrate one step and the size of that step.

A simple scheme of implementing a multirate method was tried first. The variable which would integrate to the smallest point in the time axis was integrated first based on the recommended time steps returned by the ode integrator. This scheme helped keep all the equations moving within at most the largest integration stepsize that can be taken (hmax). However, this scheme proved inadequate since the effect of one rapidly-changing variable on the projected integration step of another equation was not taken into account. Therefore, integrations frequently tended to cut down the recommended integration step, thus upsetting the flow of the program. Equations with high projected times did not cut down on the integration step until

the actual integration of the equation, by which time all the related variables had moved too far along the time axis, thus making interpolated results unreliable and inaccurate. The result of this was that it was frequently impossible to integrate the equation within the requested tolerance.

In a second scheme, the acceptability of the recommended steps returned by the ode integrator was checked. This was accomplished by actually calling the ode integrator once more at the end of each integration with the recommended stepsize. The stepsize of the second call (which could be equal to or less than the recommended stepsize) was used as a possible integration stepsize. This pseudo-call to the ode integrator changed none of the values in the program. Also, to monitor the acceptability of the projected integration step thus obtained, an estimator of the changes introduced as the other equations were integrated was added. As the changes that were introduced exceeded a certain acceptability level, the equation whose recommended stepsize thus became no longer reliable was integrated to bring it to the same time as that of the last equation integrated. Calling the ode integrator twice for each integration step (once for determining the recommended next step and once for actually integrating) was quite expensive. Therefore, another parallel scheme was tried, with basically the same structure. It used the recommended step returned by the ode integrator without rechecking it as a possible integration step but also monitored the effect of changes introduced by other variables.

Both of the above schemes produced acceptable results. Yet one of the complications that arose with the schemes above was that they were forced to take small steps. The reason was that when one equation was forced to actual integration because of the unreliability of its projected integration step, it was integrated to the same time as the last equation integrated. This had the disadvantage of occasionally causing the forced integration to take a very small integration step when the difference between the time of the last integrated equation and the equation whose recommended step had become unreliable was small. Thus the integrator was spending a large amount of time over a small interval when it was not necessary. Furthermore, once having reduced the stepsize greatly DIFSUB can only increase it slowly without introducing a lot of error. At the same time, using a smaller stepsize would not have helped the problem and using a larger one might have conflicted with the scheme of integrating first the equation with the smallest t'_{poss}.

The problem with the schemes above was that of requiring an actual integration step to be taken every time the projected integration step was felt to be unreliable. A change in the program logic was therefore undertaken and now every time that the projected stepsize was felt to be unreliable it was only rechecked. Thus the final program logic was:

1 - at the end of each integration step determine a projected possible integration step. (This can be done by doing a pseudo-integration or by simply accepting the step returned by the ode integrator as the possible integration step.)

2 - every time one of the projected integration steps is felt to be unreliable do a pseudo-integration to determine a new reliable projected integration step.

One question that has not been discussed up to now is how the program checks that a projected integration step has become unreliable. This is accomplished by having a mechanism for accumulating the worst possible case errors that might be introduced by a connected equation. The analysis is that we want to find the errors introduced to the j-th equation as a result of the i-th variable being integrated. Assuming that the i-th variable predicted value for time $(t_{curr})_j$ was y_i^0 before integration and is y_i^n after integration we want to estimate the error introduced to the j-th equation as a function of the difference between the interpolated value to the $(t_{poss})_j$ of y_i based on its value at the old $(t_{curr})_i$ and its interpolated value based on the value at the new $(t_{curr})_i$. Let y_i^0 stand for the value interpolated from the old $(t_{curr})_i$ and y_i^n stand for the value interpolated from the new $(t_{curr})_i$. Then we have for our local error estimate:

$$h_j \left| f_j(y_j^p; y_i^0) - f_j(y_j^p; y_i^n) \right| \simeq h_j \frac{\partial f_j}{\partial y_j} (y_i^0 - y_i^n)$$

Now, $(y_i^0 - y_i^n)$ is our local error estimate, had we actually integrated to $(t_{poss})_j$ point. We extrapolate this difference from the predictor-corrector difference (which is the estimate for local error) at the point $(t_{curr})_i$. Let R stand for $\frac{(t_{poss})_j - (t_{curr})_i}{h_i}$. Then the extrapolation produces:

$$(y_i^0 - y_i^n) \simeq e_i \sum_{j=0}^{k} R^j l_{j,i}$$

and therefore our local error estimate becomes

$$h_j \frac{\partial f_i}{\partial y_j} e_i \sum_{j=0}^{k} R^j l_{j,i}$$

which is

$$\delta_j = \delta_j + \left[\sum_{q=0}^{k} l_{q,i} \left[\frac{t_j^* - t_i}{h_i} \right]^q \right] e_i (t_j^* - t_j) \frac{\partial f_j}{\partial y_j}$$

i refers to the variable that is actually being integrated

j refers to the variable connected to the equation actually being integrated

k is the order of the method being used to solve the i-th equation

h refers to the actual integration step

e refers to the estimate of the local error

l refers to the coefficients of the method being used.

III - RESULTS

a - Efficiency Comparisons

The two schemes that were tested in MRATE as multirate methods were :

1 - No preintegration

In this scheme, an actual integration step would be followed by the setting of projected time to current time plus recommended time step returned by the ode integrator. A pseudo-integration step would only be taken in case the error control mechanism indicated that a certain t_{poss} had become unreliable.

2 - Preintegration

The actual integration step in this scheme would be followed by a pseudo-integration step which would test the recommended step returned by the actual integration. t_{poss} would be set to time current plus step used in the pseudo-integration. Again a pseudo-integration step would be taken in case our error control mechanism indicated that t_{poss} for a certain equation had become unreliable.

A number of different test problems were run on both of the schemes and a sizable amount of statistics was generated for each integration. The statistics gave us information on how many function evaluations were needed, how many real or pseudo-calls to the ode integrator were made, how many times corrector was called, how many of those times

it could not converge and therefore had to cut down the stepsize, how many times the converged corrector's local error estimate failed to meet the user-specified local error tolerance, how many times method had to readjust the projected integration step, and also order change data, mean stepsize and standard deviation on stepsize. Out of this abundance of data, the number of function evaluations, defined as how many times the expression $y_i'(t) = f_i(\tilde{y} ; t)$ was evaluated, was chosen as the criterion to use in evaluating the efficiency performance of the method. The decision was made because the number of function evaluations seemed to be connected to execution times and thus it offered a way of combining all the rest of the data.

The second scheme was expected to be more expensive then the first in terms of accuracy per function evaluation ratio, since the step recommended by our ode integrator should be a reasonable one unless the solution curve of the equation is changing very rapidly. We included the scheme nonetheless, to see if a better estimate of the next stepsize was needed. As is to be seen from the results that follow, the preintegration scheme did on the average 20 - 30 % worse in terms of accuracy per function evaluation than the original non-multirate DIFSUB. The no-preintegration scheme, on the other hand, consistently did two to three times better than DIFSUB and three to four times better than the preintegration scheme. For most problems, it seems that the stepsize

recommended by our integrator is accurate enough and does not justify the cost of doing additional integration steps to test the recommended stepsize.

The test problems that were tested were as follows:

1 - The system of ode's derived from the

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial t}$$

partial differential equation by forward differencing. This was tested with $u(0,t) = 0$ and different x-axis initial conditions such as:

$$u(x,0) = \frac{x}{2}$$

$$u(x,0) = \sin(x)$$

$$u(x,0) = \frac{(1-x)}{2}$$

This produced the following scheme :

$$y_1'(t) = \frac{(y_2(t)-y_1(t))}{dx}$$

$$y_2'(t) = \frac{(y_3(t)-y_2(t))}{dx}$$

.

.

.

$$y_9'(t) = \frac{(y_{10}(t)-y_9(t))}{dx}$$

$$y_{10}'(t) = \frac{(y_{11}(t)-y_{10}(t))}{dx} = -\frac{y_{10}(t)}{dx}$$

Systems with 10 and 20 ode's were tested.

2 - The system of ode's derived from the same partial differential equation with the same initial conditions.

This time centered differencing instead of forward differencing was used except for the first equation. This produced the following equations :

$$y_1'(t) = \frac{(y_2(t)-y_1(t))}{dx}$$

$$y_2'(t) = \frac{(y_3(t)-y_1(t))}{2dx}$$

$$\vdots$$

$$y_9'(t) = \frac{(y_{10}(t)-y_8(t))}{2dx}$$

$$y_{10}'(t) = -\frac{y_9(t)}{2dx}$$

Again systems with 10 and 20 equations were tested.

3 - The system of ode's made up of n of the following equations :

$$y_i' = -\left[e^{\kappa(y_i+1)(x_{i-1}-\tau)}(y_i+1) + e^{\kappa(y_i-1)(x_{i-1}+\tau)}(y_i-1)\right]$$

$$x_i' = -\mu(x_i-y_i)$$

$$(x_0=x_n)$$

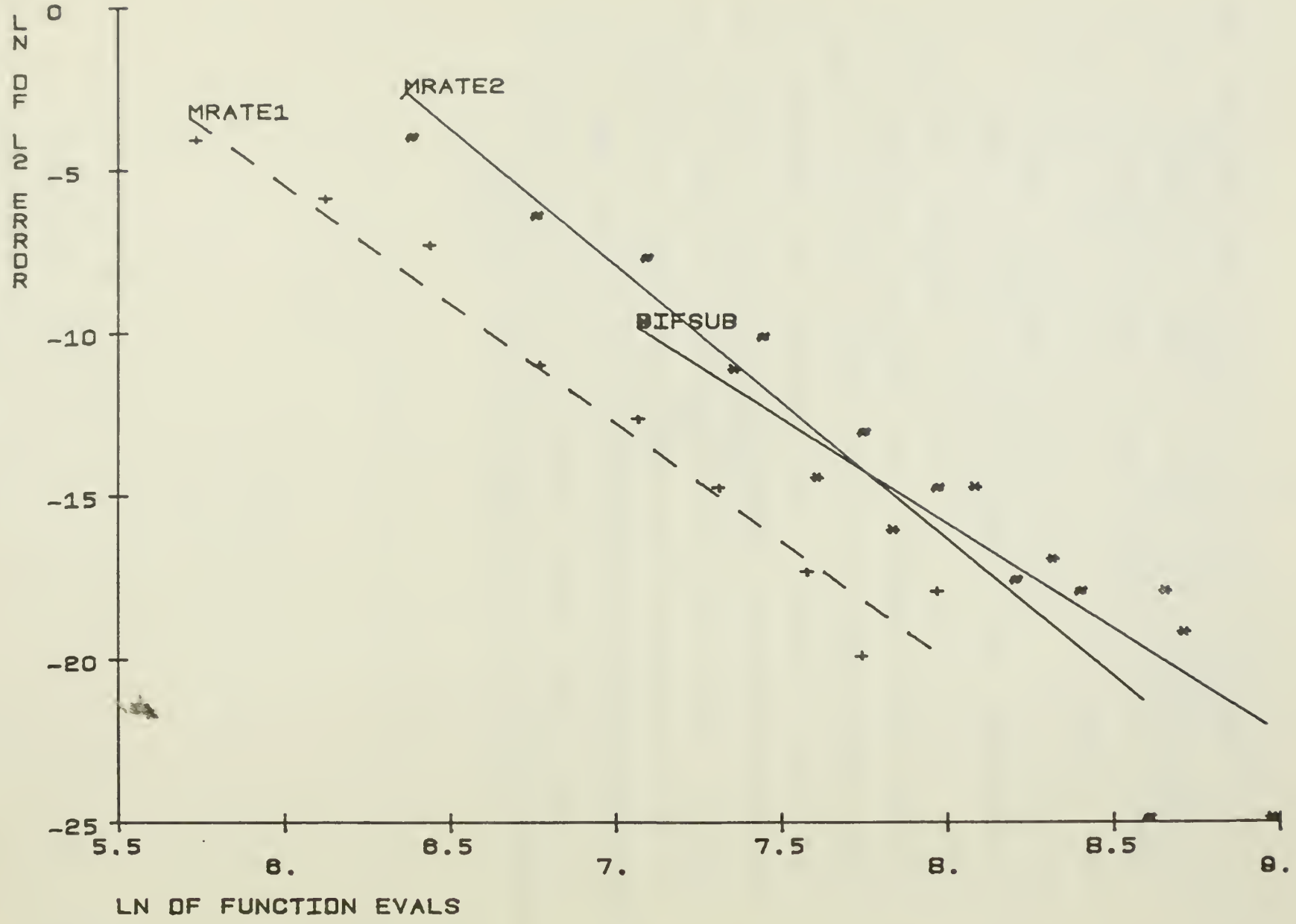
$$\mu=.5$$

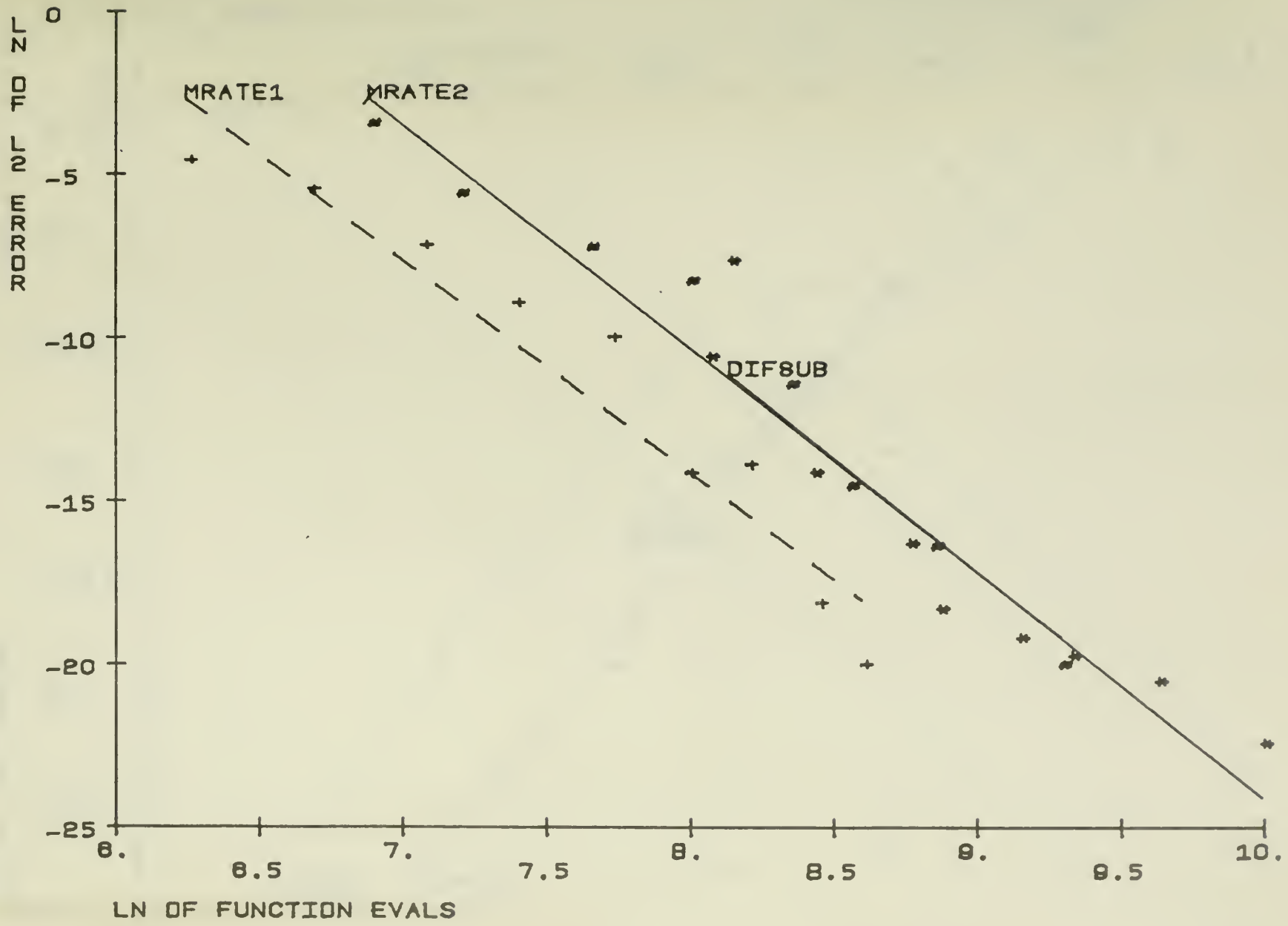
$$\kappa=20.$$

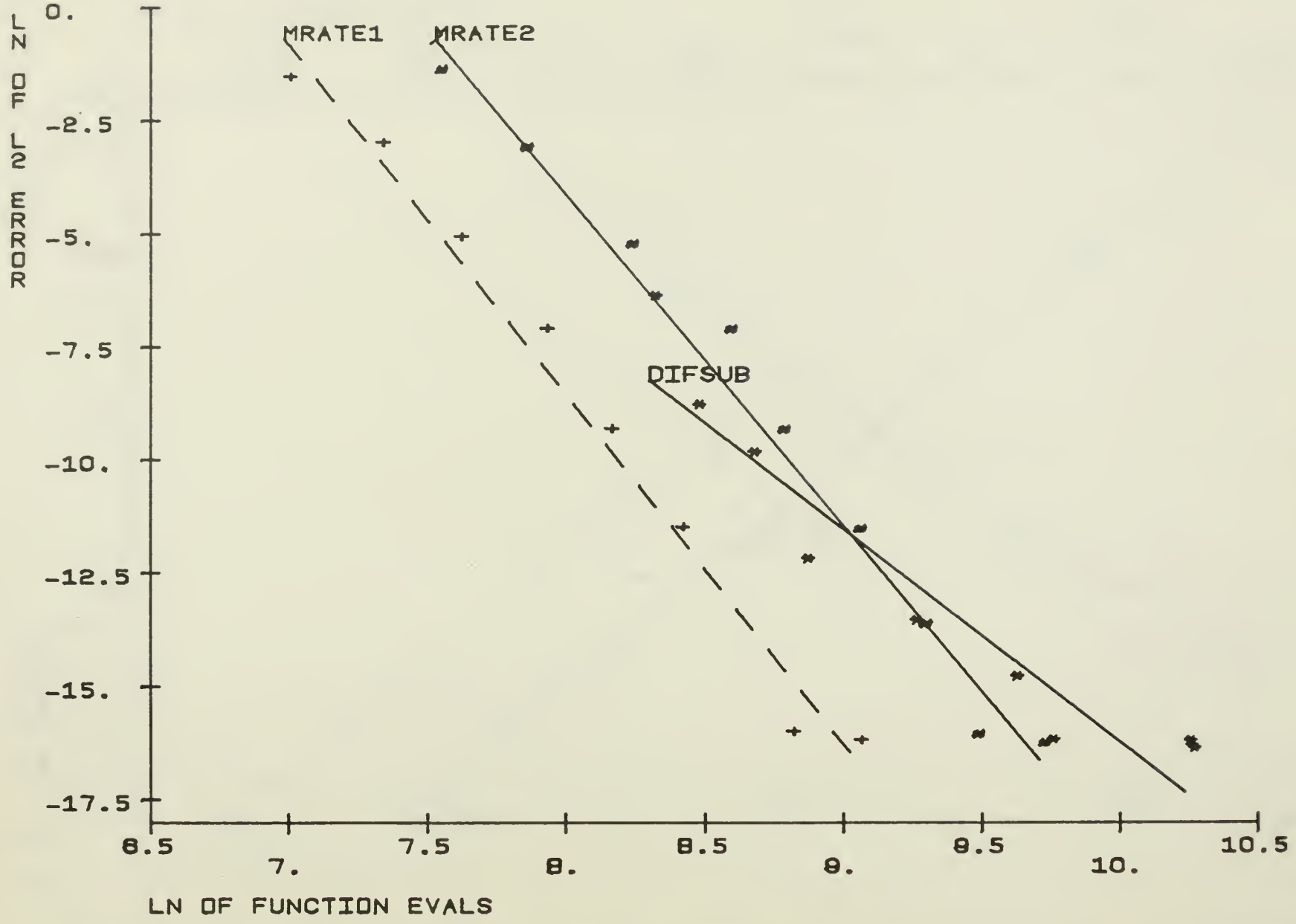
$$\tau=.5$$

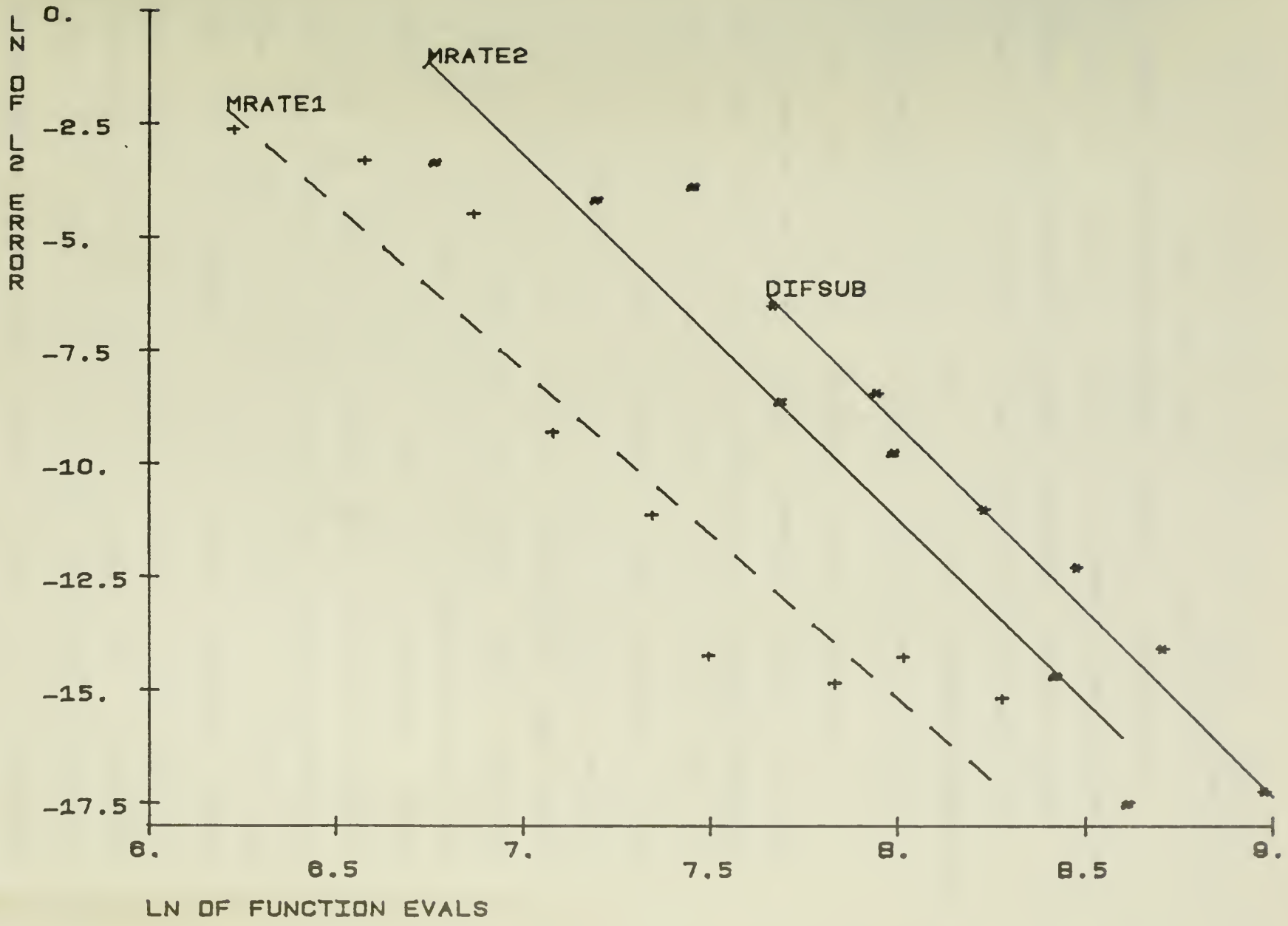
and initial conditions of $(-1,1,1,-1,-1,1)^T$ for the system of six equations and $(-1,1,1,-1,-1,1,-1,1,1,-1,-1,1)^T$ for the system of 12 equations.

Results for all of these systems were obtained and plotted. Plotting was done in terms of natural log of ||global error|| vs. natural log of function evaluations. The best fitting line in the least-squares sense was passed through all the schemes being compared, namely, no-preintegration and preintegration schemes of MRATE, and DIFSUB. The L_2 error was found by obtaining analytical solutions where possible, and otherwise by getting a true answer estimate from the values of DIFSUB employing a strict local error criterion. Graphs uniformly show superiority of the no-preintegration scheme vs. the non-multirate method and possible limited merit for the preintegration scheme vs. the non-multirate method at the high accuracy region of the graph. Some of the graphs are presented on the following pages. (In the graphs, MRATE1 refers to the no printegration scheme and MRATE2 refers to the preintegration scheme of MRATE.)


LN OF ERROR
0
-5
-10
-15
-20
-25
MRATE1
MRATE2
DIFSUB
5.5
6.
6.5
7.
7.5
8.
8.5
9.
LN OF FUNCTION EVALS


LN OF ERROR
0
-5
-10
-15
-20
-25
MRATE1
MRATE2
DIF8UB
6.
6.5
7.
7.5
8.
8.5
9.
9.5
10.
LN OF FUNCTION EVALS


MRATE1
MRATE2
DIFSUB
0.
-2.5
-5.
-7.5
-10.
-12.5
-15.
-17.5
6.5
7.
7.5
8.
8.5
9.
9.5
10.
10.5
LN OF FUNCTION EVALS


LN OF RMS ERROR
0.
-2.5
-5.
-7.5
-10.
-12.5
-15.
-17.5
MRATE2
MRATE1
DIFSUB
6.
6.5
7.
7.5
8.
8.5
9.
LN OF FUNCTION EVALS

b - Parametric Study

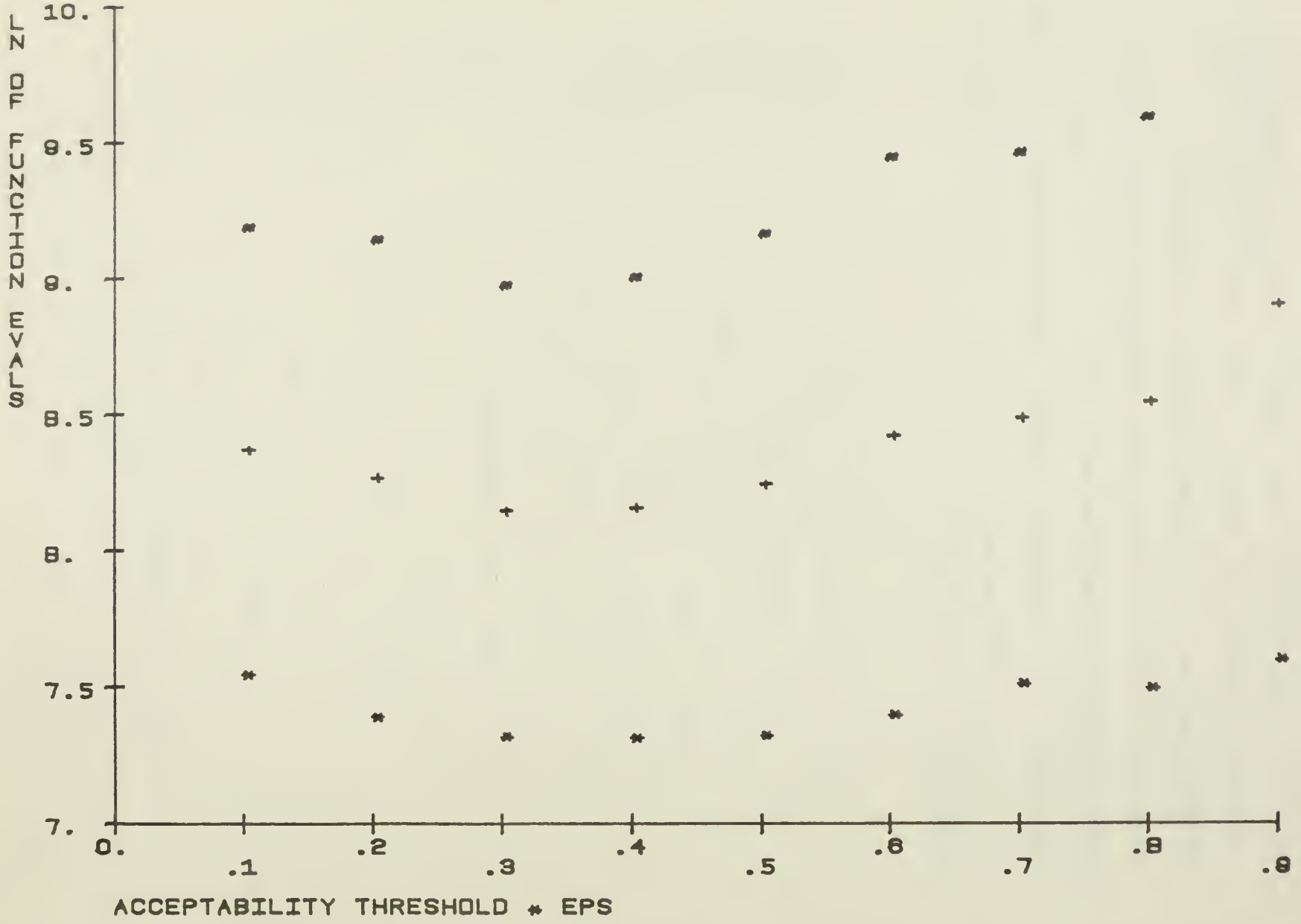
The error in a single integration step is assumed to arise from two sources :

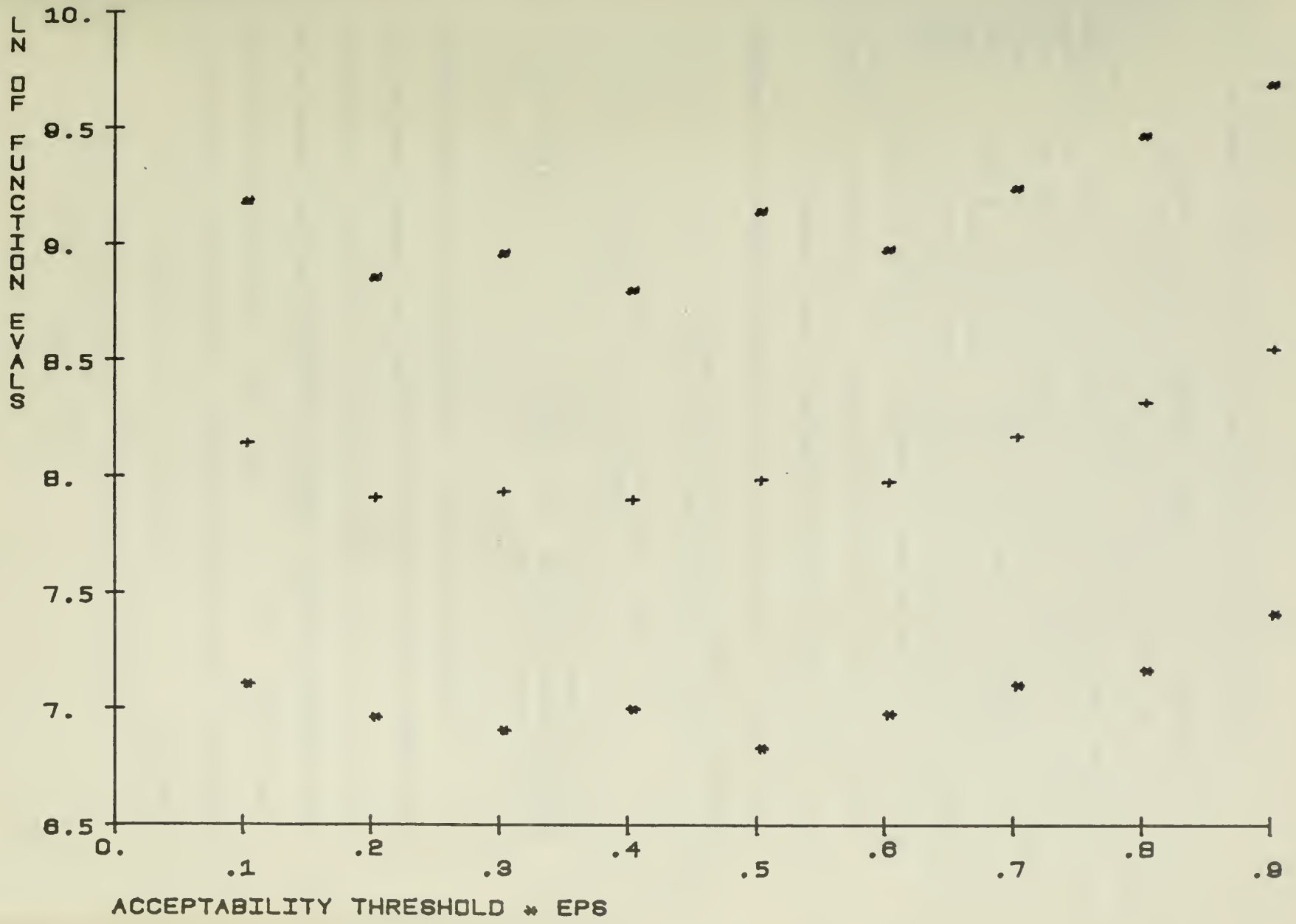
1 - Errors in the integrator applied to a particular component when the values of all other components are calculated by extrapolation,

2 - Errors introduced when coupled components are integrated.

One of the ad hoc parameters in the above was the ratio of errors "allowed" in each of those components. The original integrator recommended a stepsize which would have kept the local error at the next step just within the local error tolerance (with a certain safety margin), if the equation's solution curve had not changed radically. It was obvious that recommending as big a step as this in the multirate method would have been unwise, since that meant that our error control mechanism would report that a step had become unreliable any time a coupled equation had been integrated. Therefore, the program was modified to return a recommended stepsize which would produce an error for the next step within [0.7 * local error tolerance] . Thus our error control mechanism did not declare a stepsize unreliable until the local error estimate surpassed [0.3 * local error tolerance] . The results thus obtained were quite successful, as presented above, yet there was still room for possible improvement by a more careful testing of the above factors

which were implemented in an ad hoc manner. Therefore, a parametric study was done on the above factors to find a near-optimum set of these factors. Problem (a) of the preceding section was used with different initial conditions. Unreliability thresholds were set from [0.1 * local error tolerance] to [0.9 * local error tolerance] with corresponding changes made to integrator-returned recommended stepsizes.


LN OF FUNCTION EVALS
10.
8.5
8.
8.5
8.
7.5
7.
0.
.1
.2
.3
.4
.5
.6
.7
.8
.9
ACCEPTABILITY THRESHOLD * EPS


LN OF FUNCTION EVALS
10.
9.5
9.
8.5
8.
7.5
7.
6.5
0.
.1
.2
.3
.4
.5
.6
.7
.8
.9
ACCEPTABILITY THRESHOLD * EPS

The preceding graphs were plotted by passing the least-squares fit for each parameter to the equation

$$\ln \text{error} = \alpha(\ln \text{function evals}) + \beta$$

and then interpolating the results for three different values of global error (actual values used were 1.E-3, 1.E-6, 1.E-9) . Then for each different global error, points were plotted and the least-squares fit for each global error was passed through the quadratic equation:

$$\text{function evals} = \alpha(\text{threshold param})^2 + \beta(\text{threshold param}) + \gamma$$

(for constant global error)

and threshold parameters which minimized number of function evaluations were obtained. These were:

	problem (a)	problem (b)
global error = 1.E-3	.4414	.4075
global error = 1.E-6	.3624	.3833
global error = 1.E-9	.3439	.3656

The high accuracy results are probably more significant since results in the low accuracy region are constrained by the maximum stepsize parameter. As can be seen from this study, then, the best breakdown seems to be around (0.60 - 0.65) * eps for returned stepsize by integrator and (0.35 - 0.40) * eps for acceptability threshold.

IV - Program Usage

The implemented program, MRATE, can be used for any system of ordinary differential equations. It is most useful nonetheless, for systems of coupled ordinary differential equations which have differing speeds of integration. As such, it can be particularly suited to a variety of systems of coupled ode's obtained from discretizations of partial differential equations.

The package as it presently stands needs three pieces of information from the user.

1 - A main program to supply minimum stepsize, maximum stepsize, local error tolerance, initial time, final time, report mode, a y array containing initial values, four integer arrays to implement two lists of which variable influences which equation and vice versa, the dimension parameters for the preceding arrays, number of equations, a double precision and an integer work array. The main program need only call the package once.

2 - A function evaluator difunl. The arguments to this subroutine are :

   a - The current values of the variables

   b - The derivative (output parameter)

   c - The index of the equation being integrated

   d - The number of equations

The user need only set the derivative.

3 - A double precision function jacob to evaluate Jacobian.

Arguments are :

a - Partial with respect to this variable

b - Equation number whose partial derivative we are calculating

c - Current values of all variables

d - Number of equations

V - Further Improvements

There appear to be two areas of further research and possible improvement for the multirate package developed.

As far as the software is concerned, numerical evaluation of the Jacobian should eliminate the need for the tedious evaluation and coding of the exact Jacobian. This would be useful, since the symbolic Jacobian evaluation by hand is particularly error-prone and tedious for real problems.

Of course another possibility would be implementing this package as a part or an option of a more general differential equation integrator such as the MODEL [3] package , currently under research at the University of Illinois. Besides alleviating the necessity of supplying a Jacobian evaluation routine, this would eliminate the need to supply anything but a specification of the problem in a more natural language. If this gets implemented, it will also be beneficial to have code for discretizing partial differential equations by the method of lines available in the package [4] . The multirate method can be especially suited to such systems.

In terms of numerical analysis, there seems to be a lot of room for improvement on three different issues which warrant further research.

a - Evaluation of the Jacobian. It seems it would be sufficient to use an approximate Jacobian, until the program finds that

the approximate Jacobian is not useful any longer (i.e. when accumulated errors do not give a good estimate of the reliability of the next integration step) . It should be sufficient to update Jacobian only then.

b - Error estimates for coupled variables when an equation is being integrated. Our present scheme relies on a worst possible case analysis and assumes that the errors are the largest possible. Since our control mechanism is a pessimistic one, a possible scheme might be to keep some history of the usefulness of our acceptability criterion and change it accordingly.

c - Maximum allowable stepsize for an actual integration. It seems that a large step for a certain equation can put a connected equation so far behind in time from the first that it can no longer converge. This would necessitate backtracking and undoing the last integration which would be costly in terms of storage and execution time. Even though this has been encountered only rarely (twice out of more than 100 integrations completed successfully) , it would seem that a mechanism for determining maximun possible stepsize that can be taken during each integration, without disturbing connected equations irreversibly, would add to the efficiency of the program.

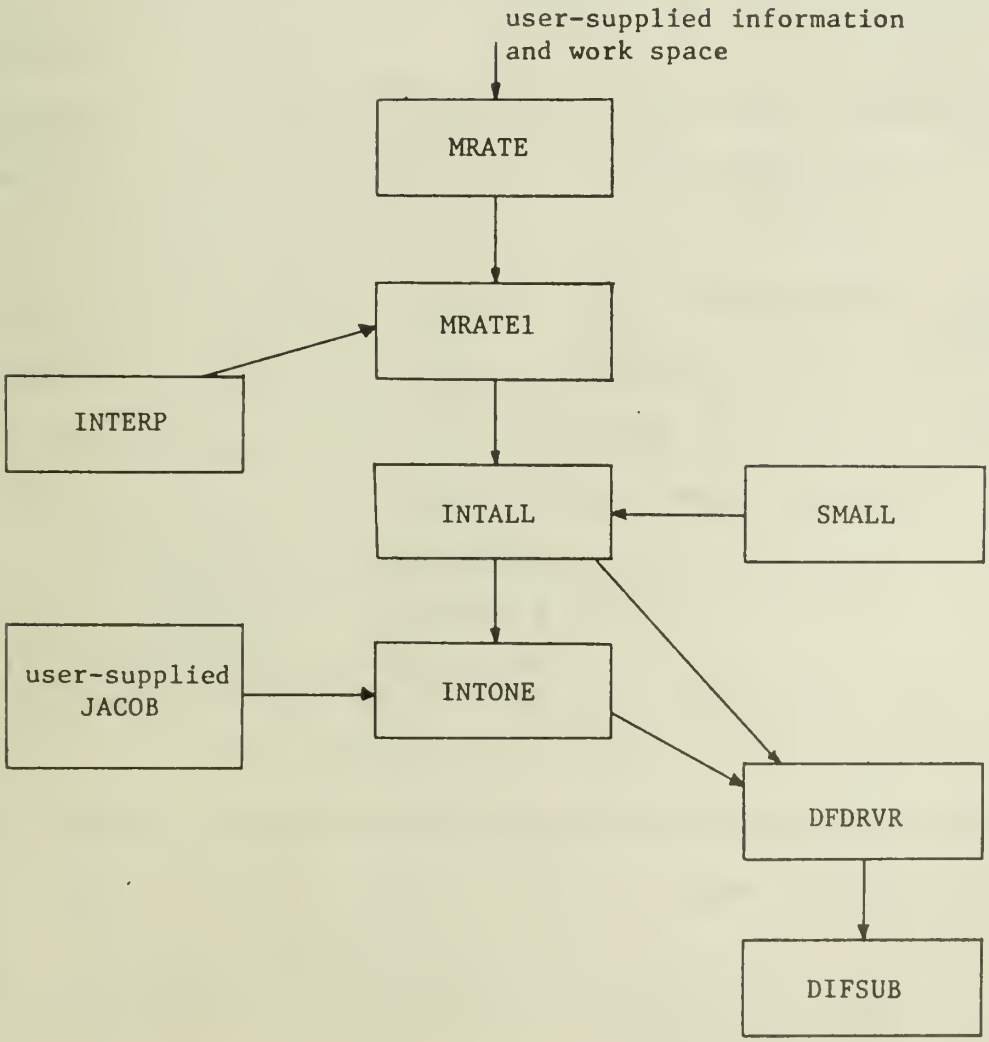
VI - Conclusion

The present study was undertaken to test the promise and potential of a multirate method for ordinary differential equations. As can be seen, the program in its present stage offers a reasonably large (2-3 times) gain in accuracy per function evaluation. This is not strictly a gain in execution times because of the overhead in implementing the multirate method (extra calls to subroutines, Jacobian evaluations, calls to interpolating routines before function evaluations), yet even though it is difficult to quantify , there still seem to be gains in accuracy per execution time. As cited in the preceding section, there seems to be a number of questions to be studied, as far as efficiently implementing a multirate method. Therefore, a multirate method has a lot of potential and seems to promise efficiency gains in solving a variety of systems of ordinary differential equations.

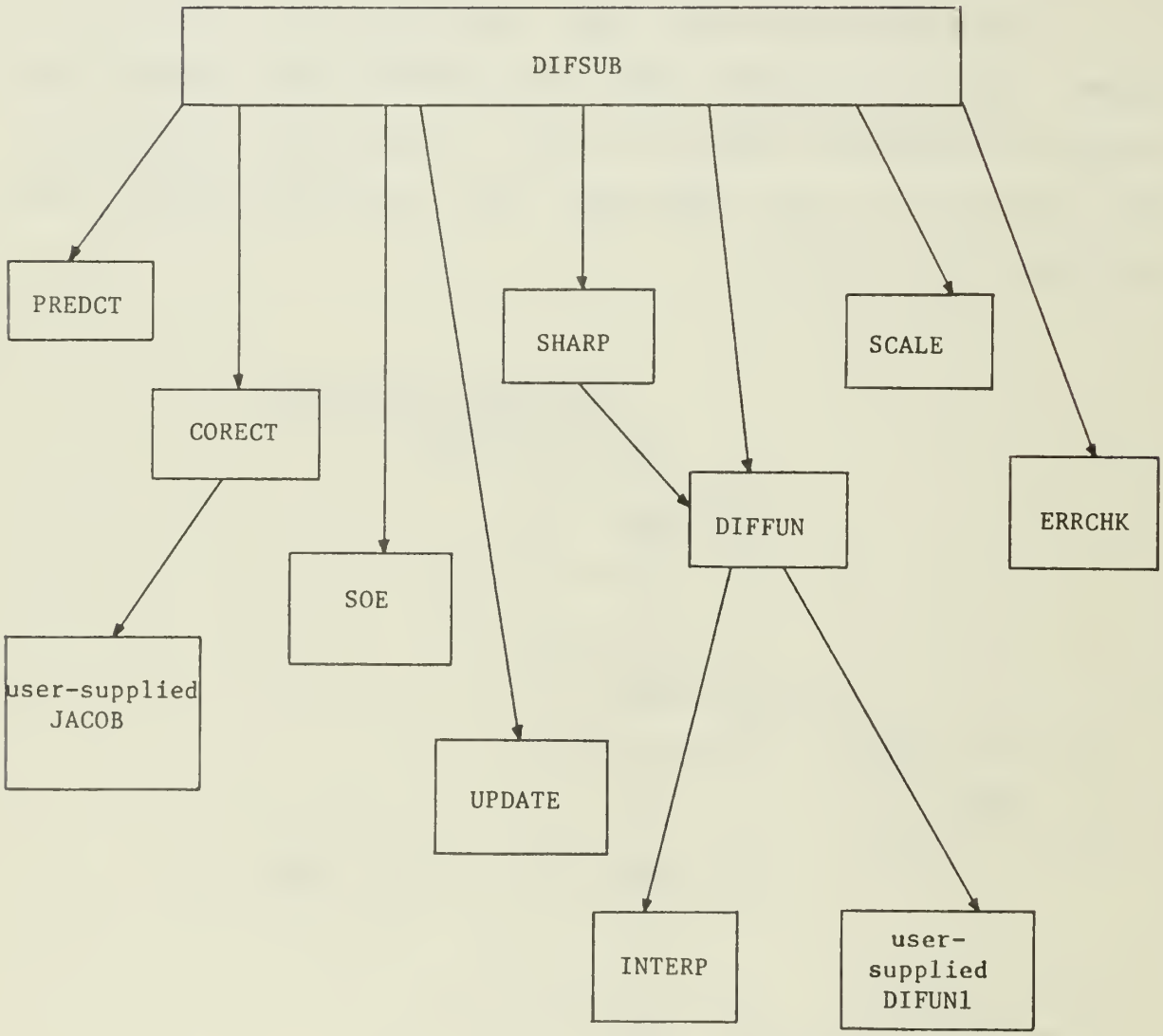
APPENDIX

I - Program

The program that has been implemented is made up of DIFSUB with subroutines that call DIFSUB and monitor the projected integration step and report integration information. The structure of the multirate ode integrator, MRATE , thus developed is as follows :



The modified subroutine DIFSUB has the following structure :



The basic changes that were made to the original ode integrator (DIFSUB) were:

1 - As can be seen above, the ode integrator has been broken down to a number of subroutines.

2 - The ode integrator now works on a single equation instead of a system.

3 - The ode integrator works on temporary arrays with its temporary results passed back to dfdrvr. Dfdrvr actually decides whether to store the temporary results (if this is an actual integration step) or to discard them (if this is a pseudo-integration).

The subroutines of the outer package accomplish :

1 - Nrate : Assigns the work space provided by the user to different arrays.

2 - Nratel : Initializes standard integration parameters. Calls intall to do the integration over the range desired and prints final values by interpolation.

3 - Intall : Main loop to do all of integration. Each loop consists of calling intone to do one integration step, then determining t_{poss} for the variable just integrated and finally determining t_{poss} for the variables whose projected integration step has become unreliable because of the effect of the last actual integration.

4 - Intone : Does one step of actual integration by a call to dfdrvr and then accumulates effect of the last integration on the coupled variables to the equation by updating delta array.

5 - Dfdrvr : Calls difsub for one step of integration and depending on flags supplied, stores temporary values produced by difsub (if it is an actual integration step) or discards

them (if it is a pseudo- integration) .

The user is requested to provide :

1 - A subroutine difunl to return the $f_i(y)$ for the i-th variable.

2 - A double precision function jacob to return $\frac{\partial f_i}{\partial y_j}$

3 - Double precision work space of 15 times the number of equations and integer work space of 5 times the number of equations in the system.

4 - A connectivity matrix of which variable influences which equation. Since systems of ode's usually have sparse connectivity matrices this has been implemented in list structure with two integer arrays.

5 - Integration parameters like minimum step, maximum step, local error tolerance, initial time and final time of integration.

Through the use of a multitude of subroutines of distinct functions, the package has become easy to change and maintain. Also since nothing on the package depends on the particular parameters of a specific problem (i.e. variable working space) it does not require any changes to suit particular problems. Finally since the program has been developed and tested on an IBM 360/75, CYBER 175 and INTERDATA 7/32, it has been written in standard, transportable FORTRAN, and should be able to run on any machine without any major modifications.

orm AEC-427
(6/68)
AECM 3201

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.
COO-2383-0056

2. TITLE
A MULTIRATE ORDINARY DIFFERENTIAL EQUATION INTEGRATOR

3. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear
Professor and Principal Investigator

Organization
Department of Computer Science
University of Illinois U-C
Urbana, IL 61801 USA

Signature Charles W Gear.

Date March 1979

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearence.
- [ ] c. Patent clearance not required.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-79-959	2.	3. Recipient's Accession No.
4. Title and Subtitle A MULTIRATE ORDINARY DIFFERENTIAL EQUATION INTEGRATOR			5. Report Date March 1979 6.
7. Author(s) A. Orailoglu			8. Performing Organization Rept. No. COO-2383-0056
9. Performing Organization Name and Address Department of Computer Science University of Illinois U-C Urbana, IL 61801 USA			10. Project/Task/Work Unit No. 11. Contract/Grant No. US ENERGY/EY-76-S-02-2383
12. Sponsoring Organization Name and Address U. S. Department of Energy Chicago Operations Office 9800 South Cass Avenue Argonne, IL 60439 USA			13. Type of Report & Period Covered M.S. Thesis 14.

15. Supplementary Notes

16. Abstracts

Most of the work on the numerical integration of systems of ordinary differential equations concerns itself with integrating all the variables simultaneously. There has been little theoretical work on integrating the variables separately (multirate methods) and even less in implementing a multirate ordinary differential equation integrator. In this thesis, a multirate integrator is presented and different schemes are tested. The implemented multirate integrator is applied to a variety of test problems and experimental results are plotted. These show the possibility of large efficiency gains in using multirate methods for a variety of systems of ordinary differential equations.

17. Key Words and Document Analysis. 17a. Descriptors

multirate integrator
ordinary differential equations

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	36
	20. Security Class (This Page) UNCLASSIFIED	22. Price

MAY 3 1979

AUG 1 1979

FOUND BY
HM

UNIVERSITY OF ILLINOIS-URBANA
3 0112 000487675